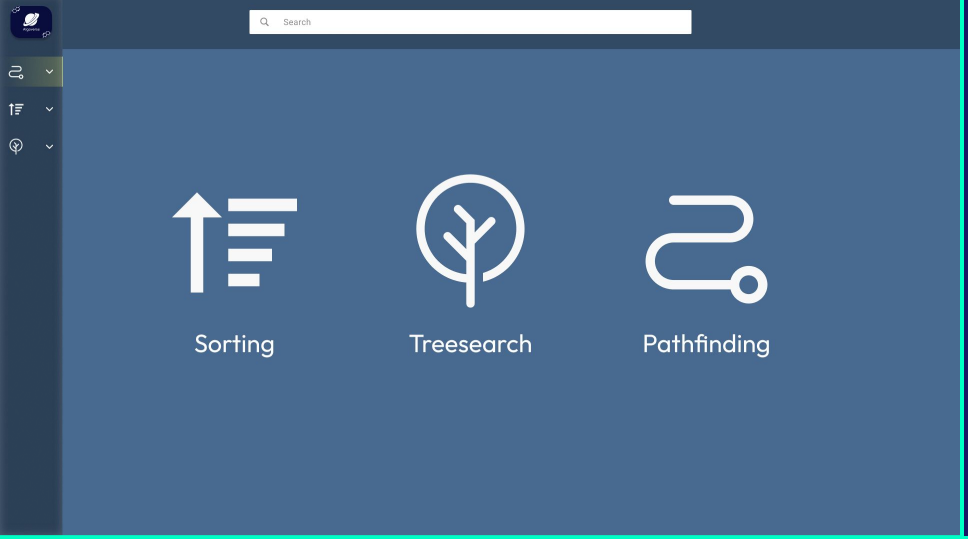
Algoverse

Bi-Weekly Progress 3

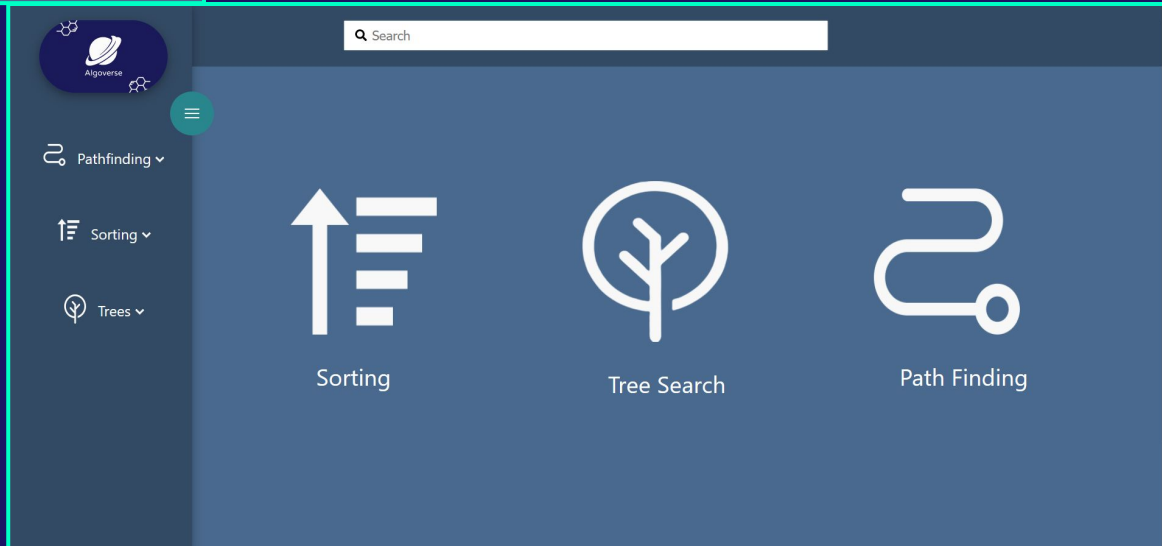
by Logic Legends

Small Recap



Design

Implementation



Order of changes

```
function BinarySearch(array, element)
  pseudocode
  pseudocode
  pseudocode
  for i < 20
    pseudocode
    pseudocode
    pseudocode
  Finish
```

```
Usecases of Binary Search:

- usecase 1
- usecase 2
- usecase 3
- usecase 4

Binary search is best used in ...
```

Algoverse

Pathfinding

- Dijkstra
- BFS
- DFS
- A*

Sorting

Trees

Shortest Path Nodes (in order):

(10, 12) (10, 13) (10, 14) (10, 15) (10, 16) (10, 17) (10, 18) (10, 19) (10, 20) (10, 21) (10, 22) (10, 23) (10, 24) (10, 25) (10, 26) (10, 27) (10, 28) (10, 29) (10, 30) (10, 31) (10, 32) (10, 33) (10, 34) (10, 35)

Useful Applications

Topological Sorting

Path Finding in Sparse Graphs

Finding Strongly Connected Components

Generating Permutations

PseudoCode

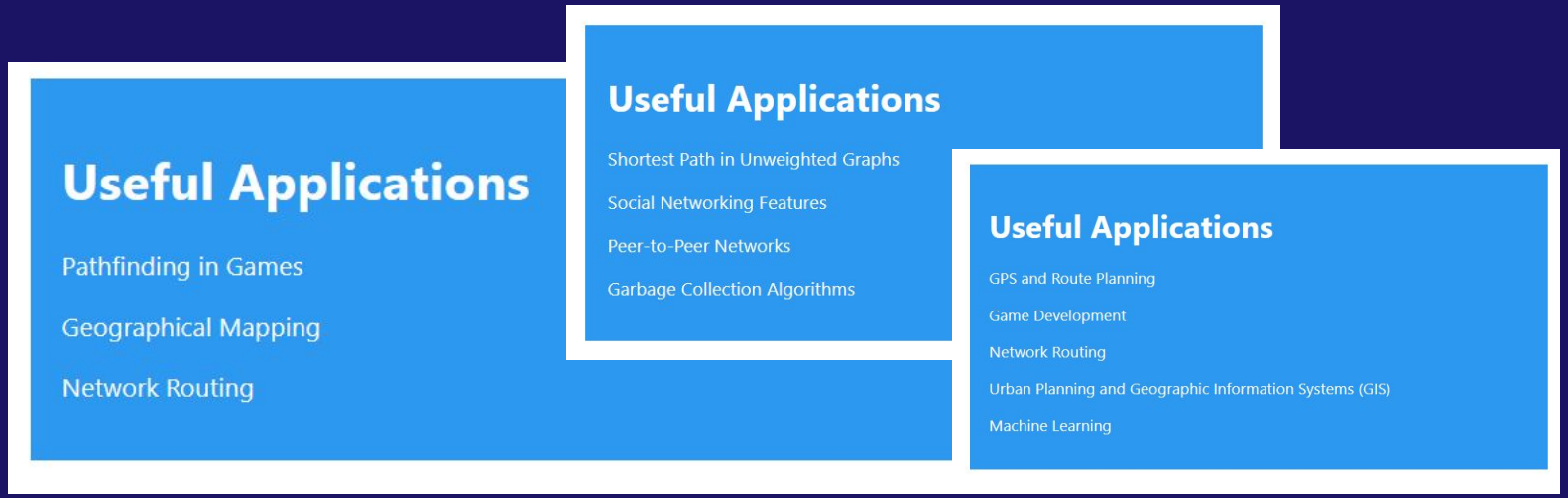
```
Start by putting any one of the graph's vertices on top of a stack.
Loop:
    1  if the stack is empty:
        1  Return or end the function.
    2  Else:
        1  Pop a vertex from the stack to select the next vertex to visit.
        2  If the vertex is not marked as discovered:
            1  Mark it as discovered.
            2  Add it to the visited list.
            3  Push all adjacent vertices (that are not marked as discovered) to the stack.
Return the visited list as the result of DFS traversal.
```

Frontend Progress – Visited Nodes

Shortest Path Nodes (in order):

(10, 12) (10, 13) (10, 14) (10, 15) (10, 16) (10, 17) (10, 18) (10, 19) (10, 20) (10, 21) (10, 22) (10, 23) (10, 24) (10, 25) (10, 26) (10, 27) (10, 28) (10, 29) (10, 30) (10, 31) (10, 32) (10, 33) (10, 34) (10, 35)

Frontend Progress – Practical Applications



Frontend Progress – Practical Applications

PseudoCode

```
Initialize Distances and Set Unvisited Nodes:
    for(vertex v in Graph):
        distance[v] = infinity
        distance[source] = 0
    unvisited = set(all vertices)
While(Processing Each Node):
    while(unvisited != null):
        current_vertex = vertex in unvisited with min distanc
        Update Distances for Neighbors:
            for each neighbor n of current_vertex:
                new_dist = distance[current_vertex] +
                edge_weight(current_vertex, n)
                if(new_dist < distance[n]):
                    distance[n] = new_dist
        unvisited.remove(current_vertex)
        Check if Finished:
            if(destination in visited or min distance in
            unvisited is infinity):
                break
Return Shortest Path:
    return distance[destination]
```

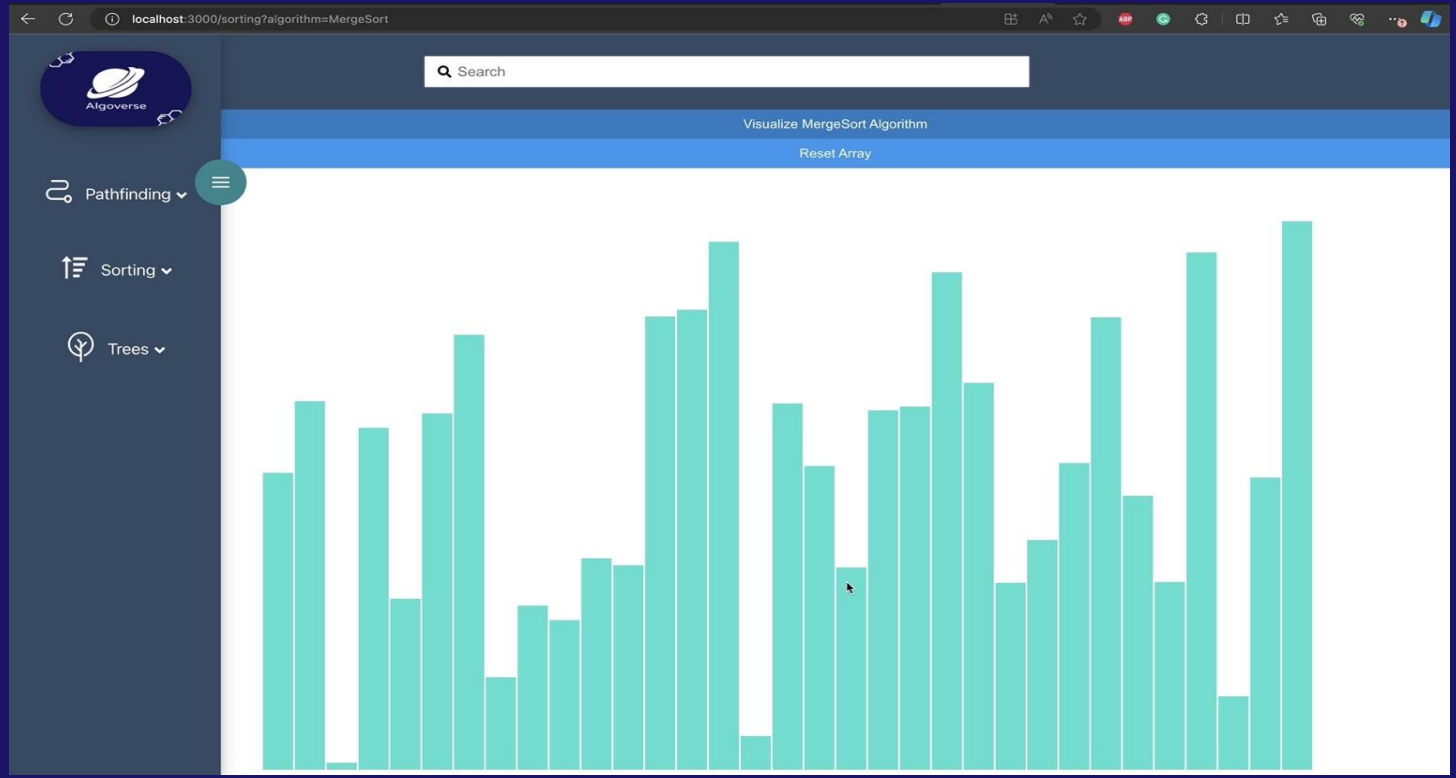
PseudoCode

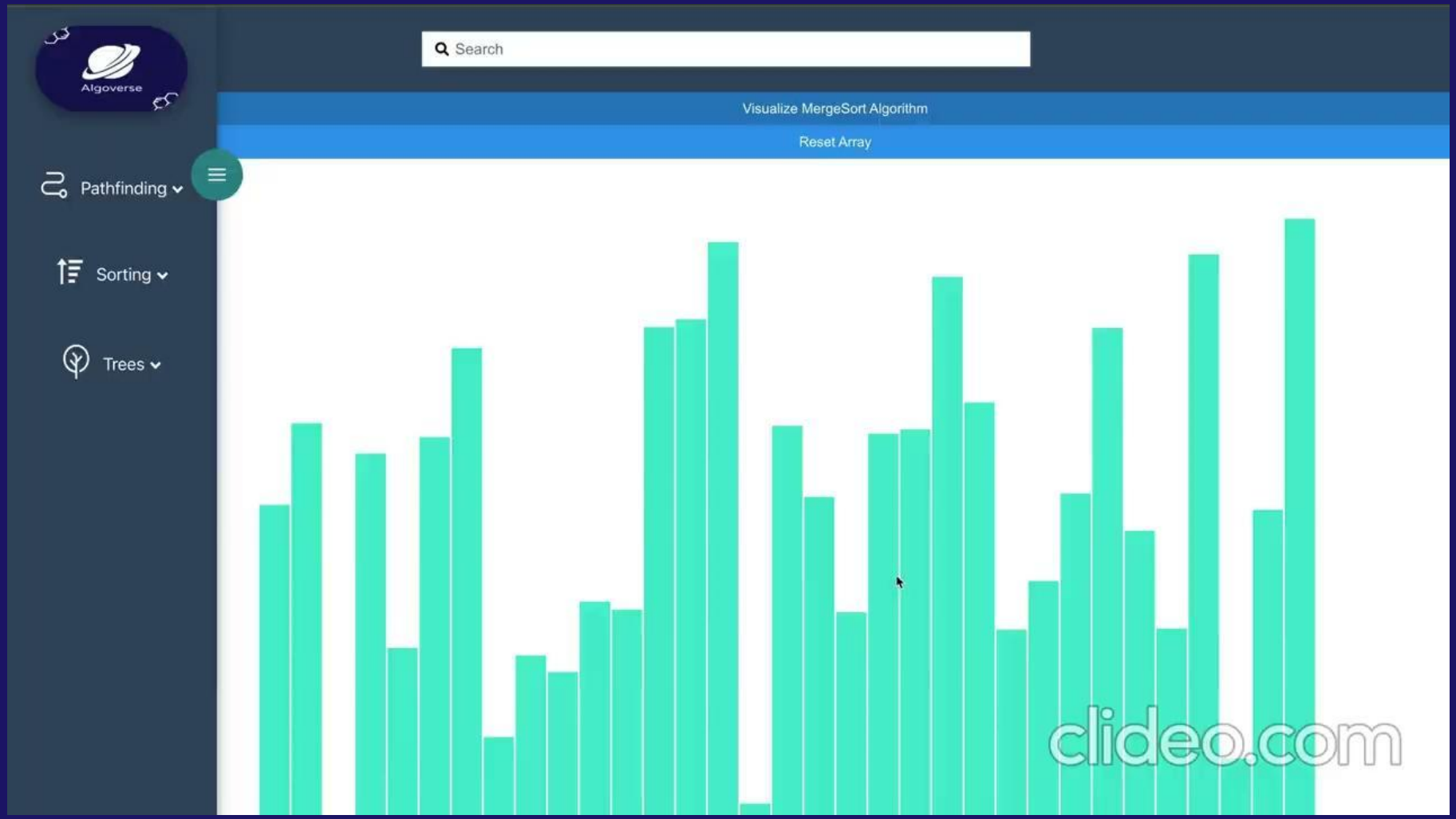
```
Start by putting any one of the graph's vertices at the back
queue.
Take the front item of the queue and add it to the visited li
Create a list of that vertex's adjacent nodes. Add the ones w
aren't in the visited list to the back of the queue.
Loop:
    if queue is empty:
        Break out of the loop.
    Else:
        Repeat the process with the new front item.
Return visited list as the result of BFS traversal.
```

PseudoCode

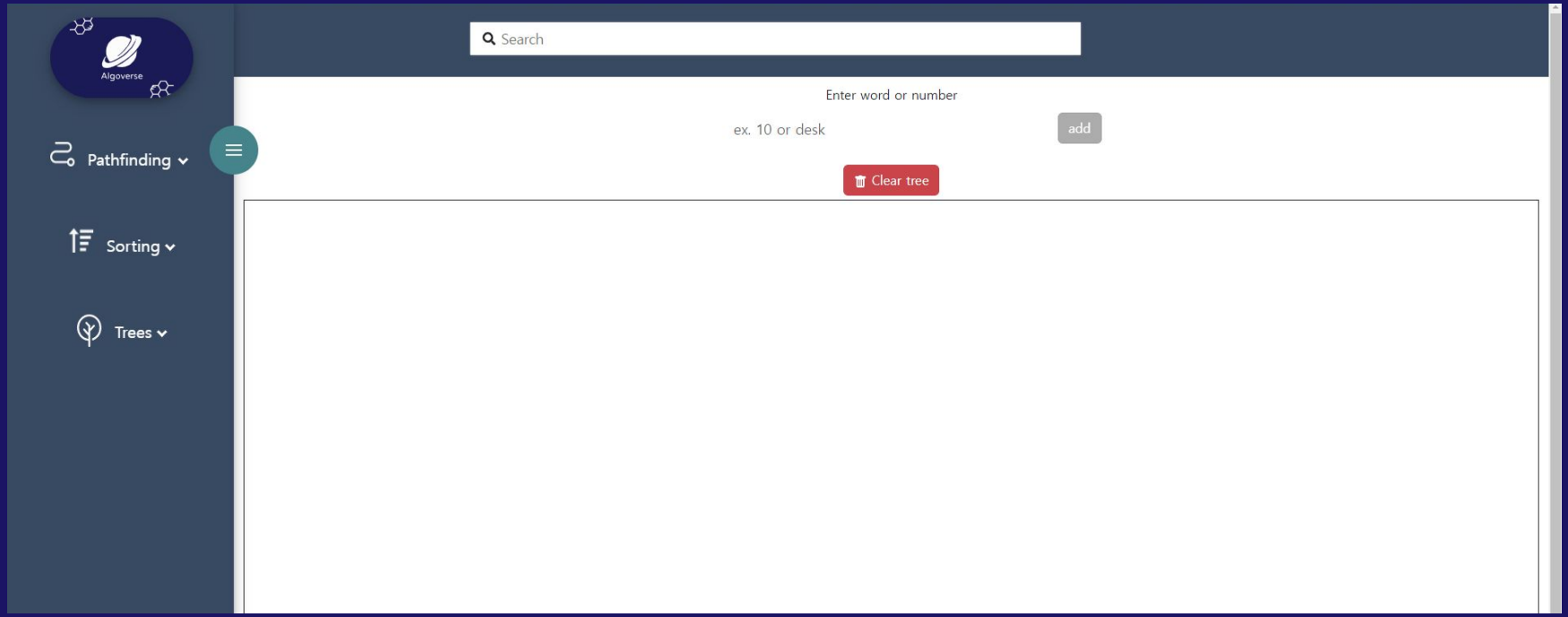
```
Start by putting any one of the graph's vertices on top of a
stack.
Loop:
    if the stack is empty:
        Return or end the function.
    Else:
        Pop a vertex from the stack to select the next vertex
        to visit.
        If the vertex is not marked as discovered:
            Mark it as discovered.
            Add it to the visited list.
            Push all adjacent vertices (that are not marked as
            discovered) to the stack.
Return the visited list as the result of DFS traversal.
```

Frontend Progress - Sorting Algorithms




Algoverse
Search
Visualize MergeSort Algorithm
Reset Array
Pathfinding
Sorting
Trees
clideo.com

Frontend Progress – Tree Search Algorithms



Algoverse

Search

Pathfinding

Sorting

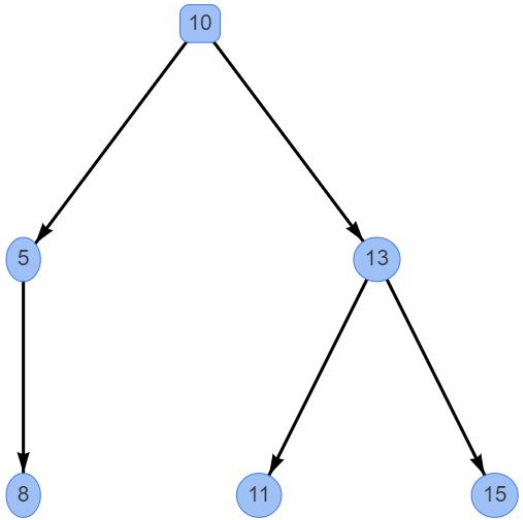
Trees

Enter word or number

ex. 10 or desk

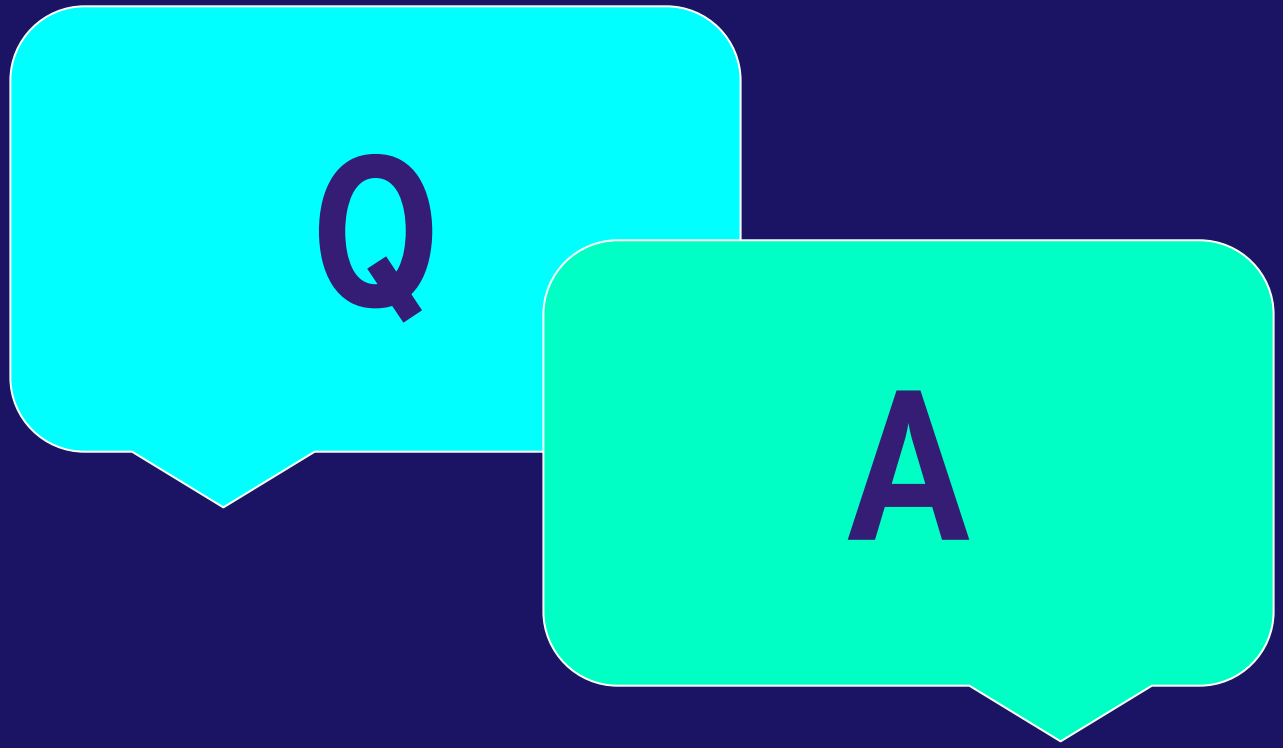
add

Clear tree



Schedule

Week	✓	✓	✓	✓	✓	12	13	14	15
Frontend	Finish Designs, Implementation Main Page + Routing		Pathfinding			Treesearch + Searchbar		Testing and Bugfixing	
Backend	Algorithm Implementation		Tests + Helping out Frontend		Buffer (more data structures, help Frontend)		Testing and Bugfixing		


Q
A